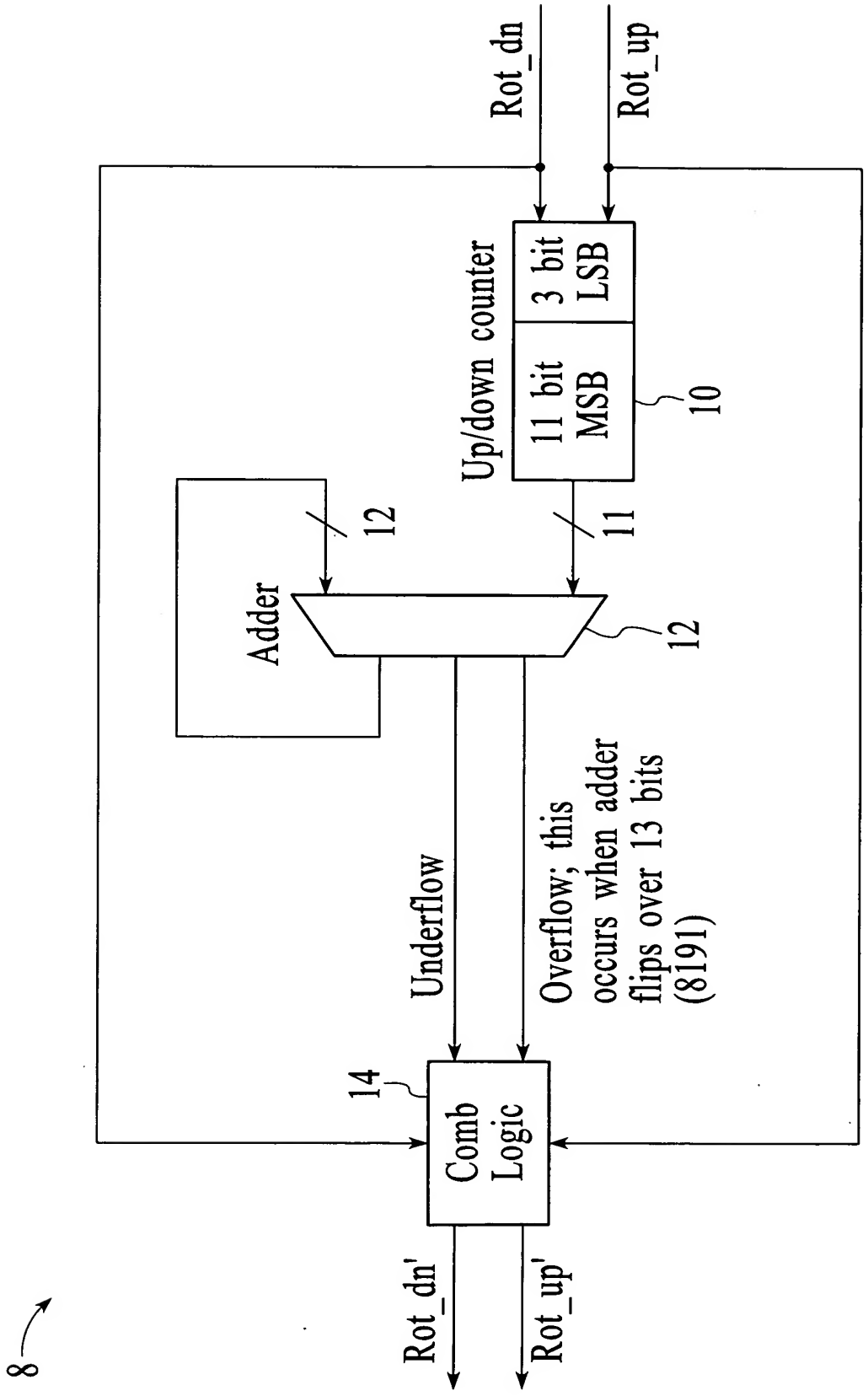
Rot_dn
Rot_up
Up/down counter
11 bit MSB
3 bit LSB
10
Adder
12
11
12
Underflow
Overflow; this occurs when adder flips over 13 bits (8191)
14
Comb Logic
Rot_dn'
Rot_up'
8

FIG. 1

Inputs				Outputs	
Rot_dn	Rot_up	Underflow	Overflow	Rot_dn'	Rot_up'
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	0	0
0	1	0	0	0	1
0	1	0	1	0	1
0	1	1	0	0	0
0	1	1	1	0	1
1	0	0	0	1	0
1	0	0	1	0	0
1	0	1	0	1	0
1	0	1	1	1	0
1	1	0	0	0	0
1	1	0	1	0	1
1	1	1	0	1	0
1	1	1	1	0	0

Equivalent value of input bits				Calculation of outputs:	
-1	+1	+1	-1	true if sum <0	true if sum >0

FIG. 2

C++ Code from system simulation:

```
//now check if the fly wheel is active                                  <- check if fly wheel is switched on
                if (FlyWheelActive)
                {
                    FlyWheelUp[FlyWheelRateCounter] = *OutUp:
                    FlyWheelDown[FlyWheelRateCounter] = *OutDown:       <- transformation of early/late to quarter rate
                    if (++ FlyWheelRateCounter == 4] // FlyWheel runs at 1/16 rate
                    {
                        FlyWheelRateCounter = 0; // Reset 0,1,2,3 counter
                        // extract the up/down overhang
                        Sum = 0;
                        for (i=0;i<4;i++)
                              Sum += FlyWheelUp[i] - FlyWheelDown[i];   <- check for early or late overhang
                        if (Sum > 0) FlyWheelCounter++;
                        if (Sum < 0) FlyWheelCounter--;                 <- step the counter accordingly up or down
                        Div_result = div (FlyWheelCounter,AverageFactor);// average over the 3 LSBs   <- Average over the LSB
                        FlyWheelAdder +/= Div_result.quot;
                        Div_result = div (FlyWheelAdder8192);// test for overflow over 2^13          <- Add the counter MSBs to the accumulator
                        if (Div_result.quot>= 1)                        <- Check for positive accumulator overflow
                        {
                              *OutUp = false;
                              *OutDown = false;
                              FlyWheelAdder = Div_result.rem;
                              Sum = *OutUp + true = *OutDown - false;   <- Generate an 'up' signal for positive overflow
                              If (Sum > 0)       'OutUp = true;
                              If (Sum < 0)       'OutDown = true;
                        };
                        If (Div_result.quot<=-1)                        <- Check for negative accumulator overflow
                        {
                              *OutUp = false;
                              *OutDown = false;
                              FlyWheelAdder = Div_result.rem;
                              Sum = *OutUp + false = *OutDown - true;   <- Generate a 'down' signal for positive overflow
                              If (Sum > 0)       'OutUp = true;
                              If (Sum < 0)       'OutDown = true;
                        };
                    };
                };
```

FIG. 3

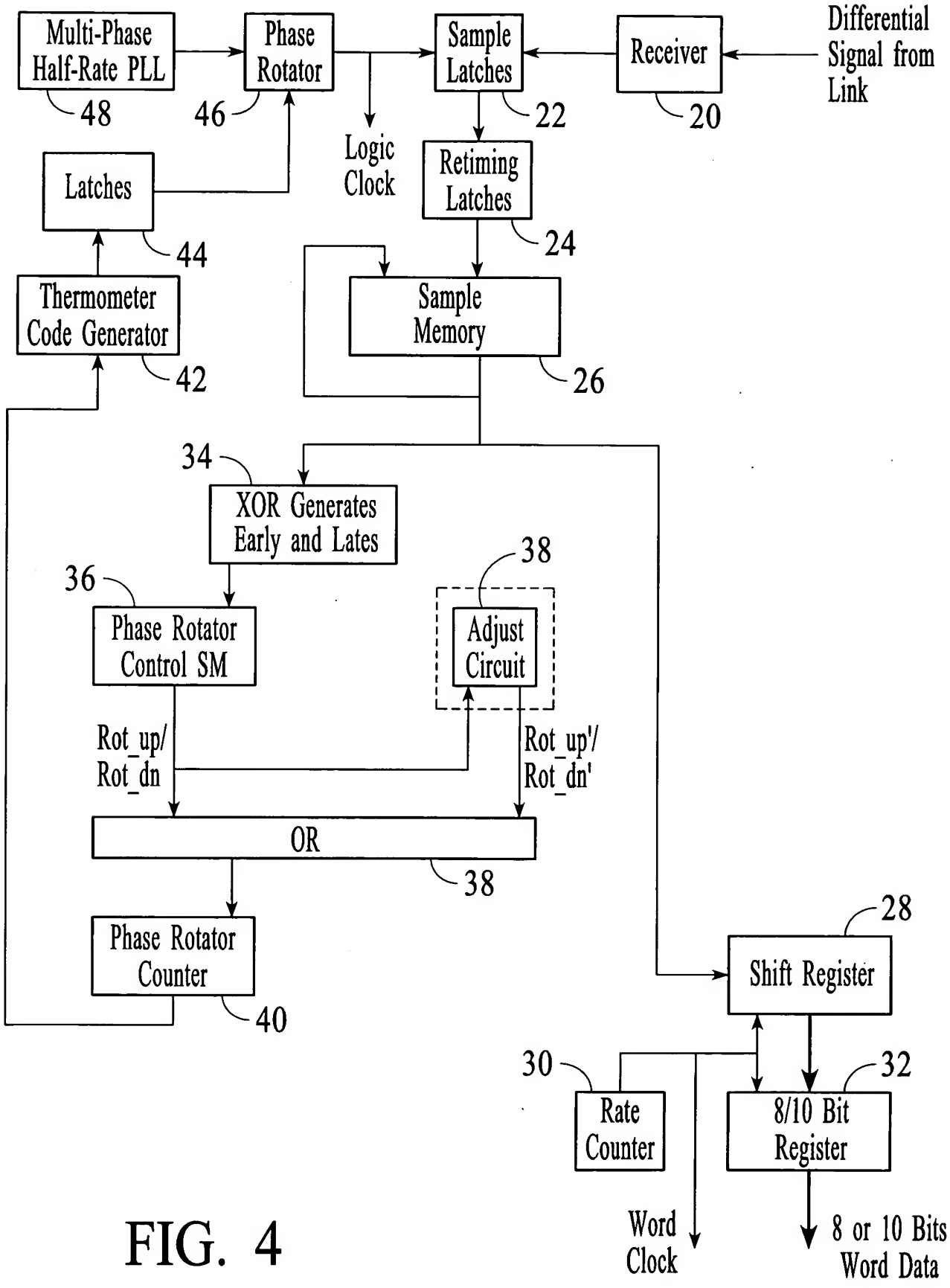
Multi-Phase Half-Rate PLL
48
Phase Rotator
46
Logic Clock
Sample Latches
22
Receiver
20
Differential Signal from Link
Latches
44
Retiming Latches
24
Thermometer Code Generator
42
Sample Memory
26
34
XOR Generates Early and Lates
36
Phase Rotator Control SM
38
Adjust Circuit
Rot_up/ Rot_dn
Rot_up'/ Rot_dn'
OR
38
Phase Rotator Counter
40
28
Shift Register
30
Rate Counter
32
8/10 Bit Register
Word Clock
8 or 10 Bits Word Data

FIG. 4